PB84-148873

DATMAN Terminal User's Guide
Version 3.0

Rhode Island Univ., Kingston

Prepared for

Environmental Research Lab., Narragansett, RI

1984

U.S. Department of Commerce
National Technical Information Service
NTIS

PB84-148873

EPA-600/D-84-032
January 1984

DATMAN TERMINAL USER'S GUIDE
VERSION 3.0

by

L. J. Bass
University of Rhode Island
Dept. of Computer Science and Experimental Statistics
Kingston, RI 02881

EPA CR 807031

EPA Project Officer
R. R. Payne

ENVIRONMENTAL RESEARCH LABORATORY
OFFICE OF RESEARCH AND DEVELOPMENT
U.S. ENVIRONMENTAL PROTECTION AGENCY
NARRAGANSETT, RI 02882

REPRODUCED BY
NATIONAL TECHNICAL
INFORMATION SERVICE
U.S. DEPARTMENT OF COMMERCE
SPRINGFIELD, VA. 22161

TECHNICAL REPORT DATA
(Please read Instructions on the reverse before completing)

1. REPORT NO. EPA-600/D-84-032	2.
3. RECIPIENT'S ACCESSION NO. PB84 148873	
4. TITLE AND SUBTITLE Datman Terminal User's Guide Version 3.0	5. REPORT DATE 1984
	6. PERFORMING ORGANIZATION CODE
7. AUTHOR(S) L.J. Bass	8. PERFORMING ORGANIZATION REPORT NO. ERLN-X94
9. PERFORMING ORGANIZATION NAME AND ADDRESS University of Rhode Island Dept. of Computer Science and Experimental Statistics Kingston, Rhode Island 02881	10. PROGRAM ELEMENT NO. C2NN1E
	11. CONTRACT/GRANT NO. CR807031
12. SPONSORING AGENCY NAME AND ADDRESS Environmental Protection Agency Environmental Research Laboratory South Ferry Road Narragansett, Rhode Island 02882	13. TYPE OF REPORT AND PERIOD COVERED Technical Report
	14. SPONSORING AGENCY CODE EPA/600/05
15. SUPPLEMENTARY NOTES	

16. ABSTRACT

DATMAN is a system supported on PRIME and PDP 11/70 computer systems which facilitates the creation, maintenance, retrieval and analysis of data. The data can be either numerical or character. Data bases can be accessed either through the terminal interface or a FORTRAN interface. For most applications the terminal interface is adequate. The FORTRAN interface is intended for more sophisticated custom applications for which the user interface is not adequate. The interfaces are completely compatible, any DATMAN data base can be accessed by either interface.

This guide describes how to use the terminal interface. Commands exist to create data bases, add data, change data, delete data, transform and retrieve data, and do analysis of data retrieved.

The analysis procedures consist of calculating descriptive statistics locally or preparing a SAS or BMDP program for submission to a remote site.

17. KEY WORDS AND DOCUMENT ANALYSIS

a. DESCRIPTORS	b. IDENTIFIERS/OPEN ENDED TERMS	c. COSATI Field/Group

18. DISTRIBUTION STATEMENT Release to Public	19. SECURITY CLASS *(This Report)* Unclassified	21. NO. OF PAGES 90
	20. SECURITY CLASS *(This page)* Unclassified	22. PRICE

EPA Form 2220-1 (9-73)

NOTICE

This document has been reviewed in accordance with U.S. Environmental Protection Agency policy and approved for publication. Mention of trade names or commercial products does not constitute endorsement or recommendation for use.

ABSTRACT

DATMAN is a system supported on PRIME and PDP 11/70 computer systems which facilitates the creation, maintenance, retrieval and analysis of data. The data can be either numeric or character. Data bases can be accessed either through the terminal interface or a FORTRAN interface. For most applications the terminal interface is adequate. The FORTRAN interface is intended for more sophisticated custom applications for which the user interface is not adequate. The interfaces are completely compatible, any DATMAN data base can be accessed by either interface.

This guide describes how to use the terminal interface. Commands exist to create data bases, add data, change data, delete data, transform and retrieve data, and do analysis of data retrieved.

The analysis procedures consist of calculating descriptive statistics locally or preparing a SAS or BMDP program for submission to a remote site.

TABLE OF CONTENTS

Page

Introduction.....1

DATMAN Data Base.....7

Data Base of Names.....9

Searchable Files.....9

JOINing Data Bases.....11

Using DATMAN System.....16

Modifying Data in a Data Base.....19

ADD.....21

INSERT.....23

CHANGE.....26

DELETE.....30

Retrieval and Transformation of Data.....32

Search.....33

Transformations.....34

Subsetting Specifications.....44

Subcommands to Search.....47

Analysis Facilities.....55

Descriptive Statistics.....55

SAS.....59

BMDP.....61

Utility Procedures.....65

BLDDB.....65

BLDFMT.....68

NAMES.....69

BLDSF.....70

INDEX.....70

Appendix A - Sample Session with DATMAN.....71
Session 1.....73
BLDDB.....73
BLDFMT.....75
ADD.....76
INSERT.....77
Session 2.....79
SEARCH.....79
DISPLAY.....79
MEANS.....79
SAVE.....80
SAS.....80
Appendix B - Data Base of Names.....82
Appendix C - List of Commands.....85

Introduction

There are three stages to the use of computers in a scientific project. They are the collection of data, the manipulation of that data, and the analysis of the data. These stages overlap somewhat and influence one another but the computer use of most projects can be divided into these categories.

The category of data manipulation or data management is the one that is usually overlooked by most non-computer professionals and yet, without some means of collating data gathered from various sources, of correcting erroneous data items and of retrieving the subset of data that is of interest at the moment, the analysis of that data frequently becomes very difficult.

It is to assist in the management of data that the computer system DATMAN was written. It provides, in a very general fashion, facilities to merge data from several sources, to change specific data items, to transform and selectively retrieve data, and facilities to assist in the analysis portion of a project.

The basic idea behind DATMAN is that data can be organized into a table. Within the table all data items in a given row are related to each other but the data on one row is independent of the data on any other row. Columns of the table are assigned names and any manipulation or retrieval of the data is done by the names of the columns. The values of the data items can be character, integer, or real.

For the remainder of this guide we will call a table of data a data base and the names of the columns will be called variable names.

Now we will go through an example which will demonstrate some of the ideas and the facilities of DATMAN. The example that we will use is that of a sample collecting series of cruises. On each cruise a variety of samples are taken at a variety of stations. These samples are then taken back to the laboratory and measured both for physical characteristics and for concentrations of several metals. An example session of DATMAN using the data base described is given in Appendix A.

When structuring a data base a critical decision is the choice of which data item defines a row in the table. DATMAN requires that one or more of the variables be used so that no two sets of values of all the chosen variables are the same. In our example one way to do this would be to assign a different identifier to each sample collected. For convenience later we choose to identify a sample by an integer value (although DATMAN allows character values). We could also have used the combination of station identifier and a local replicate number, if assigned.

Once the identification information has been decided then the next step is to list the information that is to be kept in the data base, make up variable names for that information, decide whether the information is character, integer, or a real number, and for character variables, decide the maximum number of spaces a data item can occupy.

Once we have decided on the names and the characteristics of the variables within the data base we must decide on a name for the data base and then build the data base. This is done using the program BLDDB and a sample dialogue for our example is a portion of Appendix A. The data base we are creating is called ANALYT and this name is sufficient to identify this data base to DATMAN for further manipulations.

When the samples are collected they are each given a sample number and then taken back to a laboratory for analysis. This suggests that the data for the identification for the samples in Appendix A (DATE, SNUM, STATID, LAT, LONG) should be entered into the data base at one time (when the information is known) and the data giving the measurements should be entered independently. For our example, we will enter the identification information directly from a terminal into the data base and then the measurement data will be entered separately, also from the terminal. The measurement data must be correlated by DATMAN with the identification data and we use SNUM as the mechanism to do this. Other methods besides the terminal are also useful for entering data, e.g., cards, disk files, tape files.

Since the first reference DATMAN sees to a particular sample is in processing the identification information, the new samples will, in essence, expand the table of sample numbers. This process, in DATMAN, is called an ADD. When an ADD is done DATMAN verifies that no sample numbers are duplicated. This helps control the integrity of the data in the data base.

In order to do an ADD, DATMAN must be informed of the data base to which the new data is to be added, the source of the new data, the format of the data to be added and some other utility type of information. The format of the input is specified by running BLDFMT. Appendix A includes a dialogue to create a format file for the identification information and a separate format for the measurement data.

Once the identification data has been added to the data base, the next step is to put in the measurement data. We could have added more samples than we have taken measurements and we could have added the samples in several steps. In any case, we wish to fill in columns for rows that already exist in the table. This process is called INSERT.

This is an important and potentially confusing concept. ADD is the process by which new rows are added to the data base. INSERT is the process by which data is placed in columns in already existing rows.

In order to do an INSERT, information must be given to DATMAN that is similar to that which must be furnished for an ADD. The name of the data base, the location of the input file and the name of the format file must all be furnished.

Entering data into a data base is done by a series of ADD's and INSERT's. The only sequencing requirement is the requirement that any data INSERTed must be placed into already existing rows in the data base.

Once data has been entered into a data base, most editing will be done using the CHANGE command. This is the facility provided within DATMAN for modifying specific elements in the data base. It is also possible to delete rows using the DELETE command.

Once the data has been entered into the data base and edited for correctness, the typical project will begin using the analytic capabilities of DATMAN.

These analytic capabilities are divided into two portions. The first portion of the analysis is to retrieve exactly the information necessary for the analysis and do any transformations that are necessary to put the data into the appropriate form. The second portion is the explicit analysis itself; statistical, graphical, or other.

In DATMAN, the retrieval and transformation capabilities are done using the command SEARCH. SEARCH has itself three components. First any transformations that are desired are done. These transformations can be unconditional (define a new variable as log of the sin of an old variable) or conditional (for a particular value of STATID define a new variable as 1, for other values define the new variable as the ratio of two other variables). Any transformations or new variables that are defined are, conceptually, defined for all of the existing rows in the table.

Once the transformations have been defined then the subset of the <u>rows</u> that are desired for a particular analysis are defined. The definition of the rows are done in terms of the

value of variables, e.g., include those rows for which the zinc concentration divided by the lead concentration is greater than a particular value. The specification of the rows to be retained for analysis is done with any combination of relationships and'ed or or'ed together.

After the rows to be retained have been determined, then the columns to be analyzed and the disposition of the values must be specified. The normal mode would be to list the variable names whose values are to be saved for further analysis. It is also possible at this point to a) print the values of a particular variable or combination of variables, or b) calculate descriptive statistics.

Once the desired columns of the selected rows have been saved then the analysis of the data can be done. Current analysis options are:

1) Create SAS job for submission to a remote facility
2) Create BMDP job for submission to a remote facility
3) Calculate descriptive statistics of variables broken down by categories of values of other variables

The rest of this guide is organized into detailed descriptions of the components briefly discussed here. There are the following components:

1) Definition and Manipulation of Data Bases
2) Modifying Data into a Data Base
3) Transforming and Retrieving Data
4) Analysis Facilities
5) Utility Procedures

DATMAN Data Base

A DATMAN data base can be viewed as a named table of data. The name of the table is used to identify the table for any future reference. The columns of the table have names as headings. These names (called variable names) are the only mechanism that the user has to reference particular columns. When a data base is created each variable is assigned a type. Each data item for a particular variable is interpreted according to that type. The allowable types are characters (stored internally in ASCII), integer (stored internally as 4 byte integer) and real (stored internally as 8 byte real).

The rows of the table contain the data items. Within a particular row a variable can either have a value or be undefined. Undefined values are allowed and are supported by the system but they manifest themsleves in various undesirable ways if the user is not aware of the possibility of undefined values.

No two rows of the table can be identical. In order to edit a row of data, the row to be changed must be identified. Therefore, some mechanism must be available to uniquely identify the row to be modified. In DATMAN, certain variables are specified by the user when the data base is created as being the variables to use to uniquely identify a row of data. These identification variables are called primary index variables. In some applications one variable is sufficient to uniquely identify a row, in other applications it may take more than one. Variables which are designated as primary index variables may not have undefined values.

For retrieval purposes it is desirable to be able to retrieve rows of a data base by the value of variables. In order to allow for this accessing by value, a variable can be designated as an index variable. For these variables, DATMAN will maintain the additional structures necessary to allow this accessing by variable.

Within a DATMAN data base, then, are two special types of variables. The first type, primary index variables, are used to uniquely identify a row and DATMAN maintains the integrity of these variables. DATMAN will not allow the existence of two rows with the same values of the primary index variables. The second type of special variable, index variables, are used to allow for accessing a data base by value. Primary index variables are automatically index variables.

Within DATMAN, there are certain constraints on the length of variable names and the length of character strings. The restrictions are:

1) Variable names are no more than 10 characters long. They must start with a letter, A-Z, and allowable characters are A-Z, 0-9, $, - and #.
2) No more than 7 primary index variables.
3) Character variables which are not index variables have a maximum length of 80 characters.
4) Character variables which are index variables have a maximum length of 15 characters.
5) No imitation exists upon number of variables within a data base but no command within the terminal interface will access more than 100 variables at any one execution.

Variables take up space in the data base depending upon their type. Integers take up 4 characters, real numbers take up 8 characters and character variables take up the number of characters designated as their length.

Data Base of Names

When a data base is created the information about the variables for that data base is placed into a distinct "names" data base. The name of this data base is the name of the original data base followed by DD. e.g., for data base ANALYT, the names data base is called ANALYTDD. Within the names data base it is possible to add variable names, change names or delete names using the same procedures used to add, change or delete information from a normal data base.

Caution should be used when manipulating the names data base. It is possible to destroy the integrity of a data base by making certain changes. Appendix B describes the logical structure of a data base and provides the knowledge necessary to manipulate the names data base.

Searchable Files

It is possible to take advantage of some of the retrieval features of DATMAN for fixed format ASCII files or to use DATMAN's interactive input capabilities for other applications. These uses are accomplished by the use of the "searchable file" structure of DATMAN.

A searchable file is created by the command BLDSF and it consists of two portions. One portion is the data file. This is either the name of a fixed format ASCII file or the characters TERM. The second portion of the searchable file is the name of a file describing the format of the data. The format description will be described first.

A format file (created by the command BLDFMT) contains the information necessary to describe the format of a fixed format file. That is, a list of variable names and for each variable, the type of the variable the low and high columns and the record number within a logical record which contains the data value.

This format, for example, could describe card input of fixed format or a disk file of fixed format created by another program or by the editor. The record number required allows for multiple cards to make up one logical set of data. Thus, certain variables would always be on the first card of a pair and other variables would be on the second card of the pair. It is possible for a single column to contribute to multiple variables. For example, if the date in the form YYMMDD were in col. 1-6 then DATE in col. 1-6 could be one variable and MONTH in col. 3-4 could be another.

The data file is any fixed format ASCII file where each record (card image) is no longer than 132 characters. A fixed format file is a file where the value of a particular variable is always in the same location on a record. Thus, if a certain variable is in columns 6-15 then it is always in col. 6-15. If

the data file is TERM then the user is prompted for the input at the terminal

JOINing Data Bases

The JOIN of two data bases is a logical data base (which has no physical existence) which contains all of the variables and all of the data contained in the two data bases. It can be visualized as placing the two data bases side by side. This yields a data base which has variables from both data bases and as many rows as the left hand data base. A complicating factor is how the data rows from the right hand data base are lined up with the rows from the left data base. This alignment is done by the value of the primary index variables of the right hand data base. This is perhaps best explained with an example. Figure 1 displays a data base named ANALYT. This comes from a geographical sampling experiment and is a restriction of the data base in Appendix A.

A logical problem exists with the data base displayed in Figure 1. The problem is the redundancy inherent in replicating LAT and LONG for each occurrence of STATID. When doing a retrieval it is desirable to have LAT and LONG in the same data base as STATID so that the location of a data point can be determined. On the other hand, when doing maintenance it is tedious and leads to error to change LAT for every occurrence of a particular value of STATID. The solution is to implement an operation called JOIN. This allows two physically distinct data bases (as shown in Figure 2) which for retrieval

purposes are JOINed to produce the logical data base displayed in Figure 1. For maintenance purposes each ocurrence of STATID corresponds to only one row in the second data base and this is the data base to which any changes are made.

The JOIN of the data base A with data base B is obtained by abstracting the primary index variables of data base B from the row of interest in data base A. These values are then used to logically attach the corresponding row in data base B.

How best to break apart one logical data base into several physical data bases without losing or gaining any information is a subject of research into data base systems. DATMAN allows JOINs only for retrieval and requires all of the primary index variables of the JOINed data base to be present with the same names in the data base being JOINed.

The use of the JOIN allows as a special case a hierarchical view of a data base. Each level in the hierarchy has as variables the primary index variables of the next higher level. Then by JOINing the appropriate number of levels up the hierarchy all the data in a hierarchical data base is available for retrieval while still avoiding the redundant storing of data. See Figure 3 for an example of this.

Example of Data Base Analyt

SNUM	STATID	PBCONC	LAT	MGCONC	LONG
85602	ICEA	3.56	42.365		85.62
36805	ICEB		43.423		80.854
85603	ICEA	6.28	42.365		85.62
11000	2CEA	5.8	40.832		89.63
.	.	.			.
.	.	.			.
.	.	.			.

- SNUM is used to distinguish among rows
- STATID is used to speed up retrieval

Figure 1

Physical Data Bases

DATA

SNUM	STATID	PCONC	MGCONC
85602	ICEA	3.56	
36805	ICEB		
85603	ICEA	6.28	
11000	2CEA	5.8	

LOC

STATID	LAT	LONG
ICEA	42.365	85.62
2CEA	40.832	89.63
ICEB	43.423	80.854

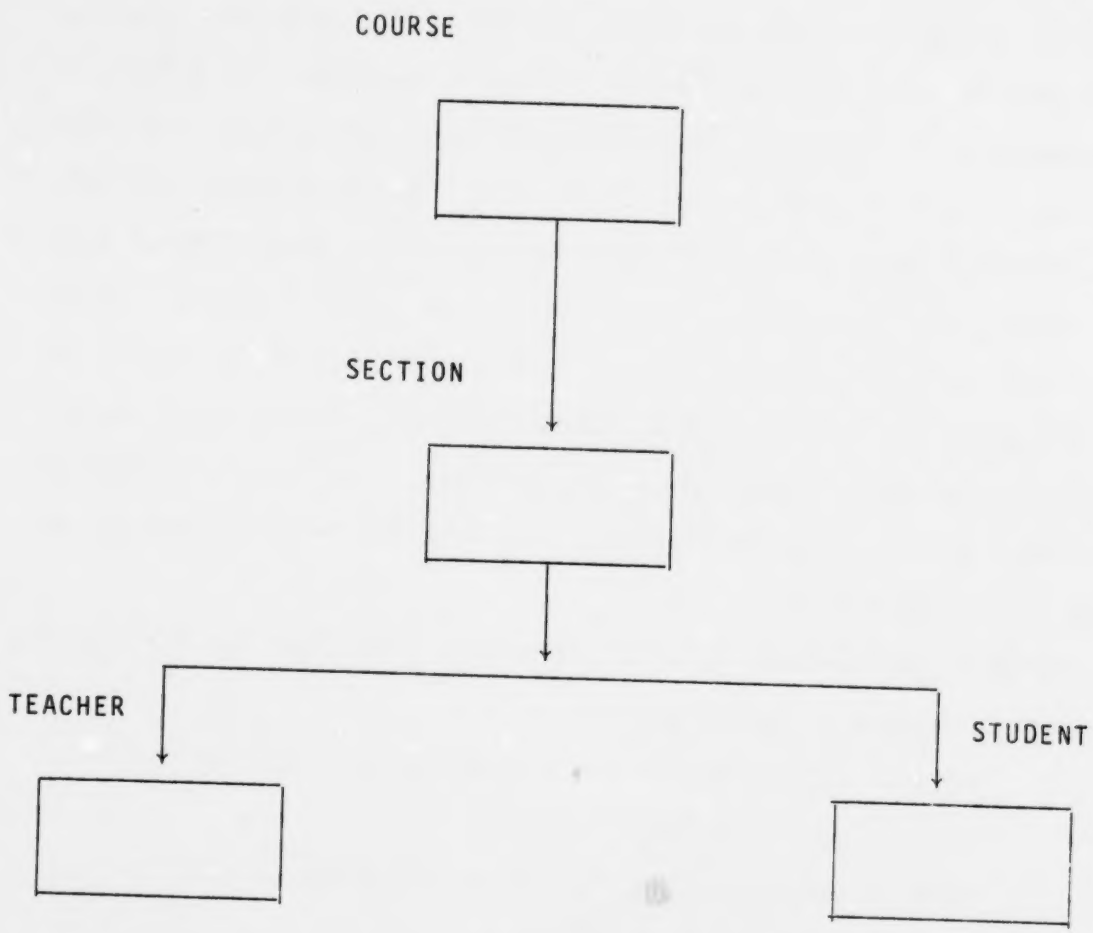
Then

DATA JOIN LOC yields

ANALYT logical data base

Figure 2

Hierarchical View of Data



In DATMAN

Separate physical data base for each category above

Then to retrieve all courses taught by a teacher use

TEACHER JOIN SECTION JOIN COURSE as a logical data base.

JOIN allows use of distinct physical data bases in hierarchical scheme while still having logical access to one path in hierarchy.

Figure 3.

Using the DATMAN System

All of the functions that are described in this guide are invoked using the program named DATMAN. The separate actions that may be used are considered either as commands to DATMAN or subcommands to SEARCH. The commands that manipulate the data in the data base, for example, are ADD, INSERT, CHANGE, DELETE, and SEARCH. When invoking a particular action the command must be furnished and then, depending upon the command, other information must be furnished. The DATMAN system allows this information to be furnished interactively, through a batch mechanism or as a combination of the two. In this section we describe how to use the system and how to take advantage of some of its features.

DATMAN has certain special commands which can be used when entering commands. These are:

HELP - This results in a descriptive explanation of current legal inputs.

STOP - This results in the termination of the current execution of DATMAN.

CLEAR - This erases all input and gives the opportunity of starting over.

DATMAN also has several pre-defined file names. These names are reserved and cannot be used for disk files accessed by DATMAN. They are:

TERM - This is the name for the user's terminal, whether input or output.

PRNT - This is the name for the printer.

Some inputs may require more than one line of input from either the terminal or the batch. In this case, the last non-blank character of any line that is to be continued must be %. The number of continuations allowed of the same line is 3. Thus the longest single input to DATMAN is 4 lines long (one line plus three continuations).

For some commands (e.g., SAVE, LIST, DISPLAY and MEANS in SEARCH) a long list of variable names may be desired. A wild card feature has been implemented which acts as an abbreviation for a list of variable names. This wild card specification allows the matching of the variable names with a user specified pattern. Those names that match are assumed part of the list.

A pattern consists of a sequence of characters (possibly none) followed by an * followed by a sequence of characters (again possibly none) followed by an *, etc. A variable name matches a pattern if the characters in the pattern occur in the variable name in the same order. Any * in the pattern is matched by any sequence of characters in the name.

For example, VAR* would match any variable name beginning with VAR, e.g., VAR1, VAR2, VAR3. VA*1 would match VAR1, VARR1, VAL1, etc. * would match all of the variable names.

Some DATMAN commands (e.g., SEARCH) work not only with DATMAN data bases but also for ASCII fixed format file. DATMAN has a structure called a searchable file which consists of a format file (created using BLDFMT) and a fixed format ASCII input file. These two file names are combined using BLDSF into a single name which is used to identify this format and input file to DATMAN. If TERM is used as the input file then DATMAN will prompt the user at the terminal for the appropriate variable values.

In any case when TERM is used as the input file (e.g., ADD and INSERT) the symbols ∂∂ as input indicate the end of the input.

After a set of responses has been given, the user is given the opportunity to change any response. After the responses are corrected, it is possible to save those responses so that they do not need to be reentered. This allows invoking DATMAN together with the saved responses by typing only one input statement. For example, when adding to the base DEMO with format file FMT1, input file TERM and no output file, those values would be given in response to questions asked by the terminal. Then those responses could be saved on a file named ADDDEMO. When adding additional data to DEMO from the terminal with the same format file and no output, DATMAN ADDDEMO will directly invoke the add procedure without the necessity of answering any questions.

It is also possible to save only a portion of the responses and to fill in the missing responses when actually invoking DATMAN. This is called making the missing values indirect. Again, using the example above, it it is desired to add data for one set of variables at one time and for another set of variables at another then two format files FMT1 and FMT2 could be created, each with its own list of variable names. Then when ADDDEMO was saved, FORMAT could have been designated as INDIRECT. In this case DATMAN ADDDEMO would result in invoking DATMAN and prompting only for the format file before the add procedure was invoked.

This ability to save all or most of a set of responses means that it becomes easier to allow non-computer sophisticates to manipulate DATMAN data bases.

Modifying Data In a Data Base

A data base may have a great many different variables and it is often inconvenient to put all of the data together in one place. Indeed, that is one of the motivations for data management systems. DATMAN makes a distinction between the creation of a new row of data (ADD) and the modification or defining of variables within an existing row (INSERT or CHANGE).

When data from a new row is entered into a data base, the table represented by the data base is extended by one row. DATMAN requires that every row have values to uniquely identify that row. Therefore, every new row must contain, at least,

values for the identification variables. Other variables need not have values entered when a row is created. ADD is the DATMAN procedure for creating new rows in a data base.

Once a row has been created then it is often desired to enter data for other variables in the row. This requires that values for the identification variables be furnished to identify the row in which the new data is to be placed. DATMAN has two separate commands to manipulate data in existing rows (INSERT and CHANGE). INSERT is intended to be used when the same variables are to be given new values for a large number of rows. CHANGE is intended to be used for a small number of changes of differing variables.

INSERT is similar to ADD in that the data to be entered must be organized in a similar fashion to the data base. That is, it must consist of rows of a table whose column names all occur as names in the data base. If the data is to come from a disk file it must be in fixed format. If the data is to be entered interactively the user is prompted at the terminal, by variable name, for the data.

Formats of data files which are not data bases are created by running the utility program BLDFMT. This program takes as input the names and types of the variables to be placed in the data base and their locations in the input file and creates a format file which is used in both ADD and INSERT. If interactive input is desired, only names and types are required.

In order to place data into a data base three pieces of information are required. These are the name of the data base, the name of the input file, and the name of the format file describing the input. Sometimes it is desired to move data from one data base to another. This is accomplished by first retrieving the desired information from the first data base (see SEARCH and SAVE) and then ADDing or INSERTing it into the second data base as appropriate.

The ADD Command

This command is used to add new rows to the data base. It takes records in one format, converts them to the format of the data base, verifies that no duplicate values are added to the values of the unique identifier and adds the new data to the data base.

The formats assumed for the input to the ADD command are incompatible with the format that the information is kept in a data base. This makes it impossible to directly add one data base to another. Formats of the ADD command, on the other hand, are compatible with the files saved during the SEARCH command. Thus, it is possible to SEARCH for the records desired, SAVE certain information from those records and use this as the input to the ADD processor. (See SEARCH).

The input to be added to the data base is organized in terms of groups of records. Each group of records in the input creates one row in the data base. This means that the input

must correspond roughly to the way the data base is set up. It is not possible for information from one group of records to be put into two rows in the data base nor is it possible for information from several groups of records to be put into one row in the data base.

If the input comes from a disk file, it is assumed to be in a fixed format. This means that if a certain variable occurs in columns 7-13 of the first record of a group then that variable occurs in columns 7-13 of the first record of subsequent groups. If it comes from a terminal the user is prompted for the values desired.

The information that describes the input is put in the format file. This includes the name of the variable, the type of the variable, the columns of the input record in which the values of the variables are found and the number of the record within the group of records that includes this variable. The utility program BLDFMT is used to create the format file (see Utilities and Appendix A). If the format is to be used exclusively for terminal input only the names and the type are necessary. No more than 100 variables can be entered through the ADD procedure.

The input to ADD consists of four statements. If the system is run interactively then the information is entered in response to prompts and the keywords (the first word in each statement shown below) does not need to be entered. If the system is run in the batch mode then the keywords do have to be on each statement and the ADD must be the first statement. The

four statements are:

1) ADD — Name where Name is the name of the data base to which data is to be added.

2) INPUT — Filename where Filename is the name of a file where the input can be found. If Filename is TERM then the input will come from the terminal in response to prompts. In this case @@ as an input will terminate the add.

3) FORMAT — Filename where Filename is the name of a format file created using BLDFMT or SAVE which describes the input format. If the input is coming from the terminal a format file must still be created giving the names and types of the variables to be entered.

4) OUTPUT — ON Filename where Filename is TERM for terminal output or PRNT for output to be printed on the line printer. In either case, the output is generated after all data has been entered. If no output is desired, omit Filename.

The INSERT Command

This command is used to place data into existing rows in a table. It takes data in one format, converts it into the data base format, locates the appropriate row of the data base and merges the data into the data base.

The information that is necessary to do an INSERT is similar to the information that is necessary to do an ADD. That is, the name of the data base, the name of the input file, the name of the format file, and (optionally) the output information. In addition, there is one other option because data is being added to existing rows.

The additional option is to allow the verification of some of the values being input. This is done by requiring that the value of one or more variables already in the row in the data base be duplicated in the input data.

The input data to be inserted into the data base is organized in terms of groups of records. Each group of records in the input creates one row in the data base. This means that the input must correspond roughly to the way the data base is set up. It is not possible for information from one group of records to be put into two rows in the data base nor is it possible for information from several groups of records to be put into one row in the data base.

If the input comes from a disk file it is assumed to be in a fixed format. This means that if a certain variable occurs in columns 7-13 of the first record of a group then that variable occurs in columns 7-13 of the first record of subsequent groups. If it comes from a terminal the user is prompted for the values desired.

The information that describes the input is put in the format file. This includes the name of the variable, the type of the variable, the columns of the input record in which the values of the variables are found and the number of the record within the group of records that includes this variable. The utility program BLDFMT is used to create the format file. If the format is to be used exclusively for terminal input only names and the type are necessary. No more than 100 variables can be referenced using the INSERT procedure.

The input to INSERT consists of five statements. If the system is run interactively then the information is entered in response to prompts and the keywords (the first word in each statement shown below) does not need to be entered. If the system is run in the batch mode then the keywords do have to be on each statement and the INSERT must be the first statement. The five statements are:

1) INSERT Name where Name is the name of the data base to which data is to be inserted.

2) INPUT Filename where Filename is the name of a file where input can be found. If filename is TERM then the input will come from the terminal in response to prompts. In this case @@ as an input will terminate the insertion.

3) FORMAT Filename where Filename is the name of a format file created using BLDFMT or SAVE which describes the input format. If the input is coming from the terminal a format file must

still be created giving the names and types of the variables to be entered.

4) VERIFY — $Name_1$, $Name_2$,... where the $name_i$ are the names of variables for which values occur in both the data base and the input. If the values for a particular row are not identical for both the input and the data base no changes are made to that row. Verification is optional.

5) OUTPUT — ON Filename where Filename is TERM for terminal output or PRNT for output to be printed on the line printer. In either case, the output is generated after all data has been inserted. If no output is desired, omit Filename.

The CHANGE Command

This command is used to change values within existing data records. In order to change a value within a data row, the particular row (or rows) to be changed must be specified and then the variable to be changed must be given a new value. It is also possible to verify the old value of the variable to be changed. This is often desirable to avoid erroneous changes. The specification of the particular records to have changes made within it is done by using values of the unique variables. This type of variable has uniqueness enforced upon it by DATMAN.

Three commands to DATMAN are necessary to invoke the change processor. Each command must be followed by at least one blank before the information portion of the command. If the system is run interactively this information is furnished in response to prompts (see USING THE DATMAN SYSTEM). These commands are:

1) CHANGE Name where Name is the name of a data base.

2) TITLE Character-string where Character-string is to be printed at the top of the report that is generated by CHANGE.

3) INPUT Filename where Filename is the name of a file where the input to the CHANGE processor can be found. Use TERM if the input is to come from the terminal.

Once the CHANGE processor has been invoked then as many changes as desired can be specified. When all of the changes desired have been made then the change processor is terminated with an END input.

The specification of the changes must contain the following information for each change to be made.

1) The row in which the changes are to be made must be identified. This is done in terms of the values of the variables used to uniquely identify a row.

2) The variable whose value is to be changed must be specified.

3) The new value of the variable to be changed must be furnished.

4) (Optionally) the old value of the variable to be changed must be furnished. This, if furnished, is used for verification. No changes are made if the old value is not as furnished on the input statement.

The general form of the CHANGE statement is:

FOR row specification CHANGE variable name

FROM $value_1$ TO $value_2$

where

1) Row specification is given in terms of some of the variables which uniquely define the rows of the data base. The simplest form of the row specification is VAR=VAL. The most general form of the row specification is:

$$VAR_1/VAR_2/.../VAR_n = LOWVAL_1 - HGHVAL_1/ LOWVAL_2 - HGHVAL_2/... /LOWVAL_n - HGHVAL_n$$

The effect of the row specification depends upon how many variables are used to uniquely identify a row. If a single variable is used to uniquely identify a row and a change to only one row is desired then VAR=value where VAR is the variable that identifies rows and value is the row with that value will specify the row to be changed. If a single variable is used to uniquely identify a row and a range of that variable will define the rows to be changed then VAR=LOWVAL-HGHVAL will change all of the rows with the value of VAR greater than or equal to LOWVAL and less

than or equal to HGHVAL. If VAR is an alphabetic variable then the range specification should be used with caution since it is not always obvious how DATMAN will decide if one character string is greater than another character string.

If more than one variable is used to uniquely identify a row then VAR_1=value still has meaning. In this case, however, there may be more than one row to be changed. All rows with VAR_1=value will be changed. The range specification can still be used with more than a single variable. If more than one variable is used to uniquely identify a row then the symbol / is used both to separate the variable names and to separate the values of those names. Thus, VAR1/VAR2=$value_1$/$value_2$ will define the row (or rows) with VAR1=$value_1$ and VAR2=$value_2$. The order of the variables does not matter but the value must occur in the same order as the variables with which they are to be associated. It is possible to use a range specification for any or all of the values.

2) Variable name - the name of the variable whose value is to be changed.

3) FROM $value_1$ - this is an optional portion of the command that verifies the current value of the variable to be changed. If the current value is not

$value_1$ then no change is made. There is no requirement that verification be done but it is desirable to help avoid erroneous changes.

4) $\underline{Value_2}$ - this is the new value to be assigned to variable name in the rows specified.

If a CHANGE STATEMENT extends past one line then place a % at the end of the line to be continued.

If a change is made to one of the variables used to uniquely identify a row then the new value is used to identify that row for subsequent changes.

If it is desired to make a value undefined then place a single . after the TO.

The DELETE Command

This command is used to delete rows from a data base. Once a row has been deleted then all of the data for all of the variables in that row is no longer in the data base.

Two commands to DATMAN are necessary to invoke the DELETE processor. Each command must be followed by at least one blank before the information portion of the command. If the system is run interactively this information is furnished in response to prompts (See USING THE DATMAN SYSTEM). These commands are:

1) DELETE Name where Name is the name of the data base from which rows are to be deleted.

2) INPUT Filename where Filename is the name of a file where input to the DELETE processor can be found. Use TERM if input is to come from the terminal.

Once the DELETE processor has been invoked then as many changes as desired can be specified. Three different types of inputs to the DELETE processor are possible. They are:

1) END - this terminates the DELETE processor.
2) ALL - this deletes all of the rows from the data base and leaves it without any values.
3) Row specification. This is the same as the row specification in CHANGE. It is given in terms of some or all of the variables which uniquely define the rows of the data base. The simplest form of the row specification is VAR=value.

The most general form of the row specification is:

$$VAR_1/VAR_2/.../VAR_n = LOWVAL_1 - HGHVAL_1/... /LOWVAL_n - HGHVAL_n$$

The effect of the row specification depends upon how many variables are used to uniquely identify a row.

If a single variable is used to uniquely identify a row and only one row is desired to be deleted then VAR1=value where VAR is the variable that identifies rows and value is the row with that value will delete the row. If a single variable is used to uniquely identify a row and a range of that variable will define the rows to be deleted then VAR=LOWVAL-HGHVAL will delete all of the rows with the value of VAR greater than or equal to LOWVAL and less than or equal to HGHVAL. If VAR is an

alphabetic variable then the range specification should be used with caution since it is not always obvious how DATMAN will decide if one character string is greater than another character string.

If more than one variable is used to uniquely identify a row then VAR1=value still has meaning. In this case, however, there may be more than one row to be deleted. All rows with VAR1=value will be deleted. The range specification can still be used with more than a single variable. If more than one variable is used to uniquely identify a row then the symbol / is used both to separate the variable names and to separate the values of those names.

Thus, VAR1/VAR2=$value_1$/$value_2$ will define the row (or rows) with VAR1=$value_1$ and VAR2=$value_2$. The order of the variables does not matter but the value must occur in the same order as the variables with which they are to be associated. It is possible to use a range specification for any or all of the values.

Retrieval and Transformation of Data

Once a data base has been built and the data entered into it, the next step in the analysis of a set of data is to retrieve and transform particular subsets of the data. This is the function of the SEARCH command.

There are three separate facets of the SEARCH command. These are the transformation facilities, the subsetting, or selection facilities, and the facilities to determine what happens to the transformed and selected data.

Before going into the details of these facilities one point about the data should be made. As should be clear from the preceding sections it is possible for certain variables in some of the rows to be undefined. This has the most impact in the SEARCH procedure. Any transformation of a variable with an undefined value will result in an undefined value. It also has an effect on whether a row is accepted for inclusion in the desired subset. If a row containing an undefined value is accepted for inclusion and that value is listed or saved then the system will output blanks for that value. These points will be clarified later. It is also possible to explicitly search for undefined values.

SEARCH

The SEARCH procedure retrieves data from a logical data base. A logical data base is either a physical data base, (created and maintained by DATMAN), a searchable file (created by BLDSF) or several physical data bases JOINed together. The JOIN of two data bases can be considered as the two data bases side by side. A variable of the logical data base is a variable of one of the JOINed data bases. See the section on JOIN for more explanation of this concept.

The input to DATMAN to start the SEARCH procedure is straightforward. We must tell DATMAN that we are searching, which logical data base is being searched, and where the input to the SEARCH processor is to be found. Again, if DATMAN is run interactively, these items will be asked for. Specifically, the input to DATMAN consists of:

1) SEARCH $\underline{\text{Name}}_1$, $\underline{\text{Name}}_2$,...,$\underline{\text{Name}}_n$ where $\underline{\text{Name}}_1$ JOIN $\underline{\text{Name}}_2$,...JOIN $\underline{\text{Name}}_n$ is the logical data base to be searched.

2) INPUT Name. Name is the name of a file where SEARCH finds its specifications. TERM if the input is to come interactively.

For example, suppose we wish to retrieve information from a data base named DEMO and the input to SEARCH is to come from a terminal. Then the input to DATMAN would be:

```
SEARCH   DEMO
INPUT    TERM
```

Once the SEARCH procedure has been invoked then the remainder of the input controls the transformation, the subsetting, and the disposition of the desired data. We will deal with these areas one at a time.

Transformations

Often it is desired to do analysis on other than the exact data in the data base. If this is the case then the data in the data base must be transformed into the form desired for the analysis. The transformation or computation facilities in

DATMAN are applied, as the user defines them, to some of the variables in each row of the table. That is, a transformation is defined in terms of the existing values of some of the variables in a data base. This transformation is applied to the values of those variables in each row of the table. No transformation can be defined which involves values from more than one row.

This last point, that transformation must be applied to one row at a time and not across rows, affects significantly the organization of the data. The choice of placing replicates, for example, on the same row of a table or in different rows of the table is influenced by the ability to average the replicates directly if they are placed on the same row.

There are two types of transformation facilities - conditional and unconditional. The unconditional transformations define computations that are applied to each row of the table. The conditional transformations are only applied to those rows that meet the test conditions. In either case, the transformations result in new variables being temporarily added to the table or new values for existing variables being temporarily placed in the table. IN NO CASE, DO ANY TRANSFORMATIONS AFFECT THE VALUES PERMANENTLY STORED IN THE DATA BASE.

The transformations are defined one at a time and are applied one at a time. Any results of the first transformation are available as input to the second transformation and so on.

Once a set of transformations are defined they are applied to the data base one row at a time.

Unconditional transformations are defined in the following way: variable=arithmetic expression. Where variable is either a name of a column already in the data base or a name of a new column to be added to the data base.

An arithmetic expression is a way of specifying what type of arithmetic has to be done to define the new value of the variable. A simple type of arithmetic expression would be to define a new value to be the sum of the values of two other variables. Parenthesis can be used to control the order of operations; that is, operations inside a pair of parenthesis are performed before operations outside the pair. Character values are enclosed in ' marks.

If it is desired to make a particular variable undefined the single character . is used. E.g., VAR1=. will make the value of VAR1 undefined.

If SEARCH is being used interactively then three special commands exist to assist the user. They are:

HELP	This produces a list of currently available variable names and functions.
STOP	This terminates this execution of DATMAN.
CLEAR	This erases all transformations entered to this point and allows starting over in the search procedure.

The valid operators and their definitions are:

Operator	Definition
1) +	x + y results in the sum of x and y as an integer if both x and y are integers and a real number otherwise. Undefined if either x or y are undefined.
2) -	x - y results in the difference between x and y as an integer if both x and y are integers and a real number otherwise. Undefined if either x or y are undefined.
3) *	x * y results in the product of x and y as an integer if both x and y are integers and a real number otherwise. Undefined if either x or y are undefined.
4) /	x/y results in the value of x divided by y as a real number. It is undefined if either x or y are undefined or if y is 0.
5) **	x**y results in x^y as an integer if both x and y are integers and as a real number otherwise. Undefined if either x or y are undefined.
6) SQRT	SQRT(x) results in the square root of x as a real number. Undefined if x is undefined or less than 0.

7) LOG — LOG(x) results in $\log_e x$ as a real number. Undefined if x is undefined or less than or equal to 0.

8) LOG2 — LOG2(x) results in $\log_2 x$ as a real number. Undefined if x is undefined or less than or equal to 0.

9) LOG10 — LOG10(x) results in $\log_{10} x$ as a real number. Undefined if x is undefined or less than or equal to 0.

10) EXP — EXP(x) results in e^x as a real number. Undefined if x is undefined.

11) SIN — SIN(x) results in sin(x) where x is in radians. Results in a real number and is undefined if x is undefined.

12) COS — COS(x) results in cos(x) where x is in radians. Result is a real number and is undefined if x is undefined.

13) TAN — TAN(x) results in tan(x) where x is in radians. Result is a real number and is undefined if x is undefined.

14) ASIN — ASIN(x) results in $\sin^{-1}(x)$. x must be between -1 and 1 and answer is a real number expressed in radians. Undefined if x is undefined, greater than 1 or less than -1.

15) INT

INT(x) results in the whole portion of x as an integer. Undefined if x is undefined.

16) ASINT

ASINT(x) results in $\sin^{-1}(\sqrt{x})$. This is the arcsin transformation. x must be positive and less than 1. Undefined if x is undefined or negative or greater than 1.

17) LOGF

LOGF(x) results in $\log_e(x!)$. This is a real number and is undefined if x is undefined. This is frequently used in calculating diversity indices.

18) INDEX

INDEX(A,B) yields the first column within the string A where the string B can be found. 0 if B is not a substring of A.

19) SUBSTR

SUBSTR(A,I,J) is the string constructed of columns I through J of the characters of the string A.

20) CONCAT

CONCAT(A,B) is the string formed by the string A immediately followed by the string B

21) JDATE

JDATE(YY,MM,DD) is the Julian date of MM/DD/YY. The Julian date starts with 1 and counts to 365 (366 for Leap Year).

22) SECND

SECND(HH,MM,SS) gives the seconds past midnight of the time HH:MM:SS. HH must be between 0 and 23.

Normally a transformation fits on one line but for long expressions it may be necessary to continue past one line. This is done by placing a % as the last symbol of the line to be continued. Then the two lines are put together and treated as if they were one, very long line. This process can be continued for up to 5 lines.

An arithmetic expression need not have any arithmetic in it. A transformation could be used to give a new name to an existing variable or to make a particular variable have a constant value. For example, if ZNCONC is a variable already defined in a data base but neither VAR1 or VAR2 are defined we could:

```
VAR1 = ZNCONC
VAR2 = 3.5
```

It is also possible to define character variables, e.g, A='ABC'.

A conditional transformation is a more powerful type of computation. It has the form:

```
IF (expression which is true or false)
    VARIABLE = arithmetic
               expression
```

The "expression which is true or false" is called a boolean expression. The meaning of the IF statement is:

1) Determine if the boolean expression is true or false. (This changes no values of any variables.)
2) If the boolean expression is false ignore the rest of the statement. (This changes no values of any variables.)

3) If the boolean expression is true, evaluate the rest of the statement as if it were an unconditional transformation statement. (This changes the value of the variable to the left of the = sign.)

The simplest form of a boolean expression is the form which tests the value of a variable. For example, A = 5. Given a value of A (from a particular row of the data base) the statement A = 5 is either true or false. Along with testing for numeric values it is also possible to test for character values. This is done by placing the value to be tested for in single quotes ('). For example, if STATID takes on alphanumeric values STATID = 'ABC' is a valid boolean expression which is either true or false for a particular row of the data base. It is also possible to test for undefined by using the single character. E.g., STATID = . will be true only if STATID is undefined for that row of the data base.

A more general test is not only for specific values but for values less than a certain value (or greater than). The most general form of a single test boolean expression is

arithmetic expression relation arithmetic expression

where the arithmetic expressions are as in the transformation statement except that they could be a character string with ' around it and the relation is one of:

1) =
2) EQ — the two values must be identical

3) <>
4) NE — the two values must be different

5) < The value on the left must be smaller
6) LT (algebraically) than the value on the right.

7) <= The value on the left must be less than or equal to
8) LE the value on the right.

9) > The value on the left must be larger than the value
10) GT on the right.

11) >= The value on the left must be larger than or equal
12) GE to the value on the right.

The above discusses the conditions for a single test boolean expression. Often it is desired to test for several conditions simultaneously. This is done through combining the tests with AND and OR. Parentheses can also be used to control the order of the operations.

The meaning of AND is that $test_1$ AND $test_2$ is true exactly in the case that both of the tests yield a true result and false if either of the tests are false.

The meaning of OR is that $test_1$ OR $test_2$ is true in case either of the tests is true and is false only if both of the tests are false.

As in unconditional transformations the IF statement is considered to occur on a single line and if it is necessary for a statement to exceed one line the symbol % must be the last

character of the line to be continued. Up to 3 lines can contribute to a single statement.

Blanks are optional in most places in the statements. It is desirable to have blanks for readability in some places. In other places blanks are required. There must be blanks before and after AND, OR, LT, LE, GE, GT, EQ, and NE. In practice it is a good idea to place blanks around the variable names and operators also. But in this case it is not required but only suggested for legibility.

As an example of the use of the transformations consider a data base with variable names of species and data as counts of the number of that particular species that occurs in an observation. Suppose we wish for each row in the data base to count the number of different species that occurred in the row. Assume we have three different species, SPEC1, SPEC2, SPEC3. Then the transformation necessary to count the number of different species are:

```
COUNT = 0
IF (SPEC1 <> 0) COUNT = COUNT + 1
IF (SPEC2 <> 0) COUNT = COUNT + 1
IF (SPEC3 <> 0) COUNT = COUNT + 1
```

When these transformations are applied to the data base we will, in effect, have added another column to the table with the heading COUNT which has the value 0, 1, 2, or 3 depending upon the number of different species present in the row.

No more than 100 variables, including both old variables and newly created variables, can be referenced in the transformation statements.

Subsetting Specification

Once the transformations have been defined then the subset of the rows that is of interest must be specified. This is done by means of the INCLUDE statement.

The INCLUDE statement chooses which rows to include based on the values of the variables in the table. All variables are treated identically during the subsetting procedure. No distinction is made between variables which are part of the permanent data base and variables which have been defined as part of the transformation process.

If subsetting is done using index variables then the retrieval will be done more quickly but this is strictly a performance and not a logical consideration.

The INCLUDE specification defines a certain subset of the rows which meet that specification. The rows are examined one at a time to determine whether the specifications have been met. Those rows that do not meet the specifications are excluded from any further processing. Those rows that do meet the specifications are passed in their entirety to the procedure processing. Thus, the procedure processing could use any of the variable names in the table and is not restricted to names that occur in an INCLUDE statement.

The statement above that "the rows are examined one at a time" does not always describe the strategy that is used for deciding which rows meet the desired specifications. If the INCLUDE statement refers to variables that are index variables

(either primary or secondary) then the number of rows that are examined is substantially reduced. This improves the performance of the transformation and retrieval function.

This means that the choice of which variables to make secondary index variables is primarily dictated by the types of INCLUDE statements that are anticipated. The INCLUDE statement has the form

INCLUDE IF boolean expression

where a blank must both precede and follow the IF.

Recall from the discussion on the conditional computation (the IF statement) that a boolean expression is a statement that is true or false based on the values of the variables in a particular row.

If the statement is true for a particular row then the row is accepted for further processing and if the statement is false then the row is rejected. The variables that are used to make up the statement can be either variables which are permanent variables of the data base or new variables which have been defined as a result of computations within the transformation portion.

A boolean expression, in its simplest form, is made up of two arithmetic expressions with some relationship between them. An arithmetic expression is a constant, a variable, or a combination of constants and variables and arithmetic operators. The relationships are <, >, =, <=, >=, <> or their character equivalent. Lists of the operators and their meaning and the relationships and their meaning are given in the

section on transformations.

A more complicated boolean expression is a collection of the simple boolean expression combined with ANDs and ORs as described in the section on transformations (IF statements).

If the results of the boolean expression are undefined (as a result of some variables being undefined or division by 0) then the row is rejected. Thus a row is accepted only if the result is true.

The INCLUDE IF statement is intended to occur on a single line. If it is necessary to extend it over more than one line the symbol % must be the last symbol of the line to be continued. No more than 3 lines can be combined in this way.

If we wish, for example, to accept those rows with STATID of CEAS and SNUM of 36201 or 38562 then we would do this as follows:

```
INCLUDE IF STATID='CEAS' AND (SNUM=36201 OR SNUM=38562)
```

If we wish, for example, to accept those rows where the ratio of zinc to silver is greater than the ratio of dry weight to wet weight then this could be done by:

```
INCLUDE IF ZNCONC/AGCONC > DRYWT/WETWT
```

A special form of the INCLUDE statement is used to retrieve the whole data base. This form is:

```
INCLUDE ALL
```

Subcommands to Search

Once the transformations and subsetting of the rows has been specified then DATMAN retrieves the rows specified and performs the calculations. If SEARCH is being run interactively the user is prompted for the commands, which are then accomplished. If SEARCH is being run in a batch mode then the commands are taken from the same input file as the subsetting commands.

The commands that are valid at this point can be found by typing HELP at the terminal. They either control future actions of DATMAN or cause an action to be immediately taken. The valid commands are:

1. LIST — generate printed output of current values
2. DISPLAY — generate terminal output of current values
3. SAVE — save current values on disk file
4. SORT — sort current values by values of specified variables
5. SEARCH — transform and resubset current values or search a different data base
6. MORE — change mode from batch to interactive
7. INPUT — change mode from interactive to batch
8. STOP — terminate this executive of DATMAN
9. DATMAN — exit SEARCH procedure and return to DATMAN
10. MEANS — calculate descriptive statistics from current values

Each command is now described more fully:

LIST

This command is used to generate a listing of values on the line printer. The format for this command is

LIST VAR_1, VAR_2,..., VAR_n; 'TITLE'

where the VAR are either the names of the variables to be printed or wild card abbreviations for variable names or both. TITLE is an optional heading for the top of the report. If included it must be enclosed in quotes.

The wild card abbreviation of variable names allows specification of a collection of variable names with one string. A wild card abbreviation has the form S_1*S_2*....*S_n where the S_i are the strings to match and the * are the wild cards. Any variable name which has the characters in S_1 followed by any other characters followed by the characters in S_2, etc. will be accepted as a variable to be listed. VAR* will list all variables starting with VAR. * will list all variables. No more than 100 variables will be listed with this command.

DISPLAY

This command is used to generate a listing of values on the terminal. The format for this command is

DISPLAY VAR_1, VAR_2,..., VAR_n

where the VAR are either the names of the variables to be printed or wild card abbreviations for variable names or both.

The wild card abbreviation of variable names allows specification of a collection of variable names with one string. * is a wild card specifier. VAR* will include all variable names starting with VAR. * will include all variable names. No more than 100 variables will be displayed with this command.

SAVE

This command saves on a disk file in card image (80 col) format the values of the variables. A searchable file is created that is suitable for use by many DATMAN commands.

The format of this command is

SAVE ON file VAR_1, VAR_2,...

where VAR_1, VAR_2 are either the names of the variables to be saved or wild card abbreviations for the names or both.

The file is the name of a disk file that is used to describe the saved data and its format. This is a searchable file and contains three physical files. These are file, file.NA and file.DA. If file is omitted it is taken to be DATKEEP.

Blanks, when they occur, are ignored except that there must be at least one blank after SAVE, one after ON, and one after file. No more than 100 variables will be saved by this command.

SORT

This command rearranges the order of the accepted rows for further processing. It has the format:

SORT BY VAR_1 BY VAR_2...

The rows are sorted in ascending order based on the values of the variables. First they are sorted based on the value of VAR_1 and any rows which have identical values of VAR_1 are then sorted into ascending values of VAR_2, etc. If it is desired to sort the rows in descending order then place (D) after the variable name. For example,

SORT BY STATID(D) BY SNUM

will order the rows in descending order by STATID and any rows that have the same STATID will be ordered in ascending order of SNUM. No more than 7 variables can be used to do the sorting.

MEANS

This procedure, without using the categorization facilities, will find the number, means, standard deviation, maximum and minimum values for a collection of variables within the current set of data. All of the rows that are input are used in doing the calculations. Any undefined values will be ignored in doing the computations. Thus, it is possible to have different numbers of observations for different variables within the same set of data. This procedure DOES NOT change the current subset being used.

The categorization facilities provide the ability to break down the overall means and standard deviations into classes based on the values of a different set of variables. For example, suppose we wished to calculate the descriptive statistics for each station where samples were collected. Then STATID would be used as a categorization variable and MEANS will produce a report giving the number, mean, standard deviation, maximum and minimum for each distinct value of STATID. It is possible to have up to seven categorization variables.

Besides calculating the descriptive statistics by categories, it is also possible to create a new data base with all of the calculated values. This provides a mechanism for the collapsing of several rows into one row for some purposes. For example, if replicates were originally stored on separate rows in a data base and, after examination of the variation within the replicates, it is desired to collect replicates into a single value. This can be done by taking means of the replicates using the appropriate identification variables as categorization variables. If these means are saved then we have a new data base in which each row now represents the means of the replicates. When this is done the categorization variables are the primary index variables of the new data base.

If it is desired to make the means be the current data values then create a new data base with the MEANS procedure and then SEARCH this new data base.

The input to MEAN consists of:

1) VAR VAR_1, VAR_2,...,VAR_n where VAR_1,...,VAR_n are either the names of the variables for which the descriptive statistics are to be taken or wild card abbreviations for them (see WILD CARD) or both. No more than 100 variables can be used in MEANS.

2) CATEG VAR_1,...,VAR_n where VAR_1,...,VAR_n are the names of the variables to be used for categorization. Up to seven of these variables may be used. No wild card specification is allowed.

3) OUTPUT ON file; TITLE where file is TERM for terminal output and PRNT for printer output. TITLE is an optional heading which will be printed at the top of the report.

4) SAVE ON File; opt where File is the name of the data base to be created with the calculated data. Opt is the list of types of data to be saved. It can be any or all of NUM, MIN, MAX, SD separated by commas. In any case the means are saved and also saved are the number of observation, Minimum, Maximum or Standard Deviations depending on NUM, MIN, MAX, or SD respectively. The names of the MEANS will be the names of the variables for each category. The names of the other variables saved are

Nxxx for number of observations of variable xxx for each category.

SDxxx For standard deviation of variable xxx for each category

MAXxxx For maximum of variable xxx for each category

MINxxx For minimum of variable xxx for each category

The categorization variables will be the primary identifiers for the data base created.

SEARCH

This set of commands is used to re-transform or re-subset the current set of values or (optionally) a different logical data base.

The format is

SEARCH data base$_1$, data base$_2$,... FROM <u>file</u>

where data base$_1$, data base$_2$, etc. are JOINed to produce the logical data base. If no data base names are specified then the current set of values is used.

<u>File</u> is the name of the input file. If it is not specified then TERM is assumed. Thus SEARCH (without arguments) will enable the transformation and subsetting of the current set of values with input coming from the terminal.

MORE

This command shifts the input mode to interactive. It is used when the input mode was originally in batch to begin a sequence of commands and then it is desired to enter additional commands at the terminal. There are no arguments to this command.

INPUT

This command shifts the input mode to batch. It is used when the mode was originally interactive and a file exists which has additional commands.

The format of this command is

INPUT file

where File is the name of the input file that has additional commands.

STOP and DATMAN

These are two commands to exit the SEARCH procedure. STOP terminates this execution of DATMAN and returns control to the operating system. DATMAN returns control to the highest level of DATMAN for other commands.

Analysis Facilities

Currently, three different types of analytic capability exist in association with DATMAN. One of these capabilities provides immediate analysis and the other two provide the ability to construct programs suitable for shipment to an IBM mainframe for execution of large standard statistical packages.

The programs that construct programs for remote execution are designed to be used with the files that are saved from the SEARCH command but can also be used with searchable files. The descriptive statistics command will work with either a data base or files saved from SEARCH. Thus, if only the analytic capability is desired for a particular set of data, it is not necessary to build a data base with that set of data. The three existing analysis capabilities are:

1) take descriptive statistics (means, max, min, standard deviation) for a set of variables within categories defined by the values of other variables,
2) create a job for execution under the SAS package, and
3) create a job for execution under the BDMP package.

We will describe these one at a time.

Descriptive Statistics

Often, the first step in an analysis is to examine the means and variation of the data across several categories. This is the function of the MEAN procedure.

This procedure, without using the categorization facilities, will find the number, means, standard deviation, maximum and minimum values for a collection of variables within a particular set of data. All of the rows that are input are used in doing the calculations. Any undefined values will be ignored in doing the computations. Thus, it is possible to have different numbers of observations for different variables within the same set of data.

The categorization facilities provide the ability to break down the overall means and standard deviations into classes based on the values of a different set of variables. For example, suppose we wished to calculate the descriptive statistics for each station where samples were collected. Then STATID would be used as a categorization variable and MEAN will produce a report giving the number, mean, standard deviation, maximum and minimum for each distinct value of STATID. It is possible to have up to seven categorization variables.

Besides calculating the descriptive statistics by categories, it is also possible to create a new data base with all of the calculated values. This provides a mechanism for the collapsing of several rows into one row for some purposes. For example, if replicates were originally stored on separate rows in a data base and, after examination of the variation within the replicates, it is desired to collect replicates into a single value. This can be done by taking means of the replicates using the appropriate identification variables as

categorization variables. If these means are saved then we have a new data base in which each row now represents the means of the replicates. When this is done the categorization variables are the primary index variables of the new data base.

The input to MEAN consists of:

1) DATA File where File is either a searchable file or a DATMAN data base where the input data is to be found.

2) VAR VAR_1, VAR_2,...,VAR_n where VAR_1,...,VAR_n are either the names of the variables for which the descriptive statistics are to be taken or wild card abbreviations for them (see WILD CARD) or both. No more than 100 variables can be used in MEANs.

3) CATEG VAR_1,...,VAR_n where VAR_1,...,VAR_n are the names of the variables to be used for categorization. Up to seven of these variables may be used. No wild card specification is allowed.

4) OUTPUT ON file; TITLE where file is TERM for terminal output and PRNT for printer output. TITLE is an optional heading which will be printed at the top of the report.

5) SAVE ON File; opt where File is the name of the data base to be created with the calculated data. Opt is the list of types of data to be saved. It can be any or all of NUM, MIN, MAX, SD

separated by commas. In any case the means are saved and also saved are the number of observation Minimum, Maximum or Standard Deviations depending on NUM, MIN, MAX, or SD respectively. The names of the MEANS will be the names of the variables for each category. The names of the other variables saved are Nxxx for number of observations of variable xxx for each category.

SDxxx For standard deviation of variable xxx
For each category

MAXxxx For maximum of variable xxx
For each category

MINxxx For minimum of variable xxx
For each category

The categorization variables will be the primary identifiers for the data base created.

SAS

SAS is the Statistical Analysis System that runs on large IBM systems. It is a general purpose program with a large number of different possible statistical analyses. DATMAN provides the facility, once the data has been transformed, subset and saved, of generating the JCL, the format description and the data cards for submission to a remote site.

Before using this facility, the user should be familiar with the use of SAS and the SAS User's Guide.

The input to the procedure in DATMAN consists of four separate categories of information. First the information to construct the JCL for the remote site must be specified. This includes the name of the user, and the name of a disk file which contains the accounting information and the information necessary to invoke SAS at the remote site. The default for the JCL will specify the JCL necessary to run SAS at the Washington Computer Center using the EPA Narragansett account and routing information. For different sites the defaults should be different. In any case, if a disk file is created with all of the JCL necessary to invoke SAS except for the user's name then the name of this disk file must be furnished to the SAS option of DATMAN.

The second category of information necessary for the procedure is the name of the searchable file which contains the data. DATMAN takes the data from a data file and format information from a format file (see BDLFMT in UTILITY section) and transforms them into a form compatible with SAS.

The third category of information is the description of the statistical analyses that SAS is to follow with the data. This information is placed on a file (possibly the terminal) and is copied into the appropriate portion of the SAS program that is generated. The variable names used by SAS will be variable names included in the format file.

The final category of information is what is to happen locally. DATMAN creates a disk file with the SAS program on it. The name of the disk file must be furnished. It is possible to concatenate several SAS jobs together for one submission to the remote site and DATMAN must be informed if this is desired.

The SAS procedure is a part of DATMAN and so the DATMAN program must be invoked first. The input to DATMAN is presented in the batch version, if DATMAN is run interactively then this information is asked for by prompts. After the first piece of information the remaining information can be placed in any order.

1) SAS your name — Your name is placed on the job card for identification.

2) INDAT filename — The name of the searchable file where the data is to be found. Default is DATKEEP.

3) OUTPUT filename($\frac{A}{N}$) — The name of the disk file where the completed SAS program, ready for remote submission, is to be placed. If this is to be the

only program currently on the file then use option N. If it is to be added to the end of the current contents of the file use option A. The default for the option is N and for the file name is SAS1.

4) JCL filename — The name of a file where the JCL to run SAS at the remote site is to be found. The JCL is placed at the front of the generated program and should be completed except for the name on the job card. Default is WCCSASJCL.

5) SASPROG filename — The name of a file where the actual SAS procedure cards are to be found.

An example of the use of this procedure is given in Appendix A.

BMDP

BMDP is a general statistical analysis package that runs on large IBM systems. It is a collection of related statistical programs with a large number of different possible statistical analyses. DATMAN provides the facility, once the data has been transformed, subset and saved, of generating the JCL, the format description and the data cards for submission to a

remote site. Before using this facility, the user should be familiar with the use of BMDP and the BMDP User's Guide.

The input to the BMD procedure in DATMAN consists of four separate categories of information. First, the information necessary to construct the JCL for the remote site must be specified. This includes the name of the user, the name of a disk file which contains most of the JCL and the number of the particular BMDP program to be used.

The JCL file contains JCL including accounting and routing information. The default contains JCL necessary to run BMDP at the Washington Computer Center using the EPA Narragansett account. For different sites the defaults should be different. In any case, if a disk file is created with all of the JCL necessary to invoke BMDP except for the user's name and the name of the particular program, then the name of this disk file must be furnished to DATMAN.

The particular BMDP program which is to be run is given in the chapter describing the procedure in the BMDP User's Guide. The program name will be BMDPxx but the chapter heading will contain only the xx and DATMAN needs only the xx.

The second category of information necessary is the data and format information. DATMAN takes the data from a searchable file and transforms them into a form compatible with BMDP.

The third category of information is the paragraphs within BMDP that describe the procedures the particular program of BMDP will follow. This information is placed on a file

(possibly the card reader) and is copied into the appropriate portion of the BMDP program that is generated. The variable names used by BMDP will be variable names included in the format file.

The final category of information describes what happends locally. The BMDP program is created on a disk file, suitable for submission to a remote site. It is possible to concatentate several programs to be submitted together.

The BMD procedure is part of DATMAN and so DATMAN must be invoked before using BMD as the first command. The input to DATMAN is presented as if the program is being run as a batch program, if DATMAN is being run interactively then this information is asked for by prompts. After the first piece of information the remaining data can be in any order.

1) BMD your name — Your name is for identification of the output.

2) PROG xx — xx is the name of the BMDP procedure that is to be used. The actual program name is BMDPxx but only the xx need be furnished.

3) TITLE characters — An arbitrary title for the BMD reports.

4) INDAT filename — The name of the searchable file when the data is to be found. Default is DATKEEP.

5) OUTPUT filename($\frac{A}{N}$)

The name of the disk file where the completed BMDP program, ready for submission, is to be placed. If this is to be the only program currently on the file then use option N. If it is to be added to the ends of the current contents of the file use option A. The default for the option in N and for the file name is BMD1.

6) JCL filename

The name of a file where the JCL to run BMDP at the remote site is to be found. The JCL is placed at the front of the generated program and should be complete except for name on the job card and the xx after the BMDP. Default is WCCBMDJCL.

7) BMDPROG

The name of a file where the actual BMDP paragraph cards are to be found. Only those sentences following the blank sentence in the /VARIABLE paragraph are to be included. Typically, only paragraphs after /VARIABLE need be included.

8) BLANK(MISSING / ZERO) — MISSING if blanks are to be treated in analysis as MISSING values, zero if they are to be treated as 0. Default is MISSING.

Utility Procedures

The definition of a data base, the constructing of the format description and the listing of the variable names within a data base are all functions that simplify the use of DATMAN. These utility functions manipulate the files that control DATMAN and do not depend upon any data in the data base. The utility functions currently available are:

1) Creating a data base
2) Building a format file
3) Listing the names of an existing data base
4) Building a searchable file
5) Creating index files for data base

Creating a Data Base

Before physically creating a data base, the structure of the table must be defined. This means that the variable names must be chosen and the types and length of each variable must be decided upon. The allowable types are R for real, I for integer, and A for character. Once a variable has been

assigned a certain type it retains that type for the life of the data base and the type is applied to each data item entered for that variable. Thus, if a variable has some real values (with a decimal point) and some integer values (without a decimal point) it should be given a type of R.

Once the variable names are listed then the particular variables whose values are used to identify the rows must be decided upon. Often this is a simple process. A particular variable, such as sample number, is designed to identify the other data items in a particular row. Other times, no single variable in the list of desired variables is sufficient to uniquely identify the other values of data items in a row. In this case, these variables must be furnished to the procedure. For example, neither the date nor the station identifier are, by themselves, sufficient to identify a particular row of the example data base (Appendix A). The combination, however, of date, station identifier, and a local replicate number would be sufficient.

The other decision that must be made prior to the actual definition of the data base is which variables are to be used to speed up the SEARCHing program. If a variable is to be used frequently as a component of an INCLUDE statement then it may be worthwhile to specify this variable as a secondary index.

A secondary index is a variable which the SEARCH procedure can use to speed up the retrieval process. The system maintains a list for each one of these variables that may allow the SEARCH procedure to decide which rows satisfy the INCLUDE condition without actually having to examine each row. This is automatically true of the unique identification variables.

The existence of these index variables introduces delays into the ADD and INSERT functions but speeds the SEARCH function for INCLUDE statements that use these variables. No necessity exists to have any of these secondary variables.

DATMAN maintains a brief description of each variable so that when the variable names are listed (see NAMES), this description can be printed. As part of the creation process, then, this description must be furnished for each variable. It is useful to include in the description things like the units of the values or other assumptions that are made about the values.

The actual creation of a data base is done by using the command BLDDB. Each data base must be given a name and any file with this name will be destroyed. Also, files name.NA, name.DA, and name.DX will be created and any existing files with those names will be destroyed.

Appendix A gives an example of creating a data base.

Once a data base has been created it is possible to modify the names or characteristics of the variables. This should be done with caution, however, since it is possible to destroy the integrity of the data base. Appendix B provides the information necessary to modify the names or characteristics of a data base.

Building a Format File

A format file is required as input to the ADD procedure, the INSERT procedure and BLDSF. It is, as the name implies, a description of the format of a particular set of data. This section will discuss the data as if it is on cards but the concepts are the same if the data is on a disk file or a tape file.

The data must be organized as if it were in a data base. That is, it must be organized in rows, and each row must occupy the same number of cards. Also, the data must be in fixed format. That is, if a certain variable is in columns 1-3 of card 1 and there is one card per row, then it must be in columns 1-3 of all subsequent rows.

To define a format for each variable, the following information must be given:

1) variable name
2) low column of variable on card
3) high column of variable on card
4) card number within cards for particular row
5) type of variable (A, R, or I) for alphanumeric, real or integer.

The concept of a card number requires some explanation. If there is one card per row then this is one. If there are several cards per row (in which case the data must be: cards for row 1, cards for row 2, etc.) then it is the card number within a particular row.

Within the fixed format constraint, it is possible to have the same column as a portion of more than one variable. For example, suppose CR1ST1REP1 is in columns 1-10, then columns 1-10 can be one variable, 1-3 another, 4-6 another and 7-10 another.

To build a format file, the command BLDFMT is used. The program will ask for the name of the file to be created and any existing file with that name will be destroyed. This name is the one that is used as input to ADD, INSERT and BLDSF.

An example of the running of this program is given in Appendix A.

If the format is to be ued exclusively for interactive input the columns and card numbers are unnecessary.

NAMES Utility

For existing data bases, it is possible to get a list of the current variable names, their types, lengths, and comments. This is done by using the NAMES command within DATMAN.

If DATMAN is run in the batch mode the input is

NAMES data base

If DATMAN is run interactively then respond NAMES to the request for the command.

Creating Searchable Files

The SEARCH procedure and the analysis procedures will work on searchable files as well as DATMAN data bases. A searchable file is an ASCII fixed format file together with a format file which describes the data file.

The format file is created using BLDFMT and the data file is created by the user. The command BLDSF will create a searchable file with a user specified name. If the data file is TERM then the input will come from the terminal and the user will be prompted for the input by variable names.

The input for BLDSF is:

1) BLDSF file — where File is the name of the searchable file to be created

2) FORMAT file — where File is the name of the format file created by BLDFMT

3) DATA file — where File is the name of the data file.

Indexing Data Base

Occasionally, it becomes necessary to recreate the index files for a DATMAN data base. This could occur when a new index variable is added to the data base, when the index category of a variable is changed or when the integrity of the data base is damaged through a system crash or improper closing with the FORTRAN interface.

INDEX is the command which recreates the index files for a data base.

APPENDIX A - Sample DATMAN Runs

This appendix will follow a data base through the following steps:

a) building data base

b) adding data into data base

c) retrieving data from data base

d) taking means of retrieved data

e) passing retrieved data to SAS

The data base to be used is based on a geographic sampling project where cruises pick up samples at various stations and take the samples back to the laboratory for various analyses.

For this example we will be dealing with the following set of data.

DATE	SNUM	CRSID	STATID	LAT	LONG	PBCONC	MGCONC
801020	85602		ICEA	42.365	85.62	3.56	
800918	36805		ICEB	43.423	80.854		
800918	85603		ICEA	42.365	85.62	6.28	
801020	11000		2CEA	40.832	89.63	5.8	
801104	55701		2CA3	40.35	88.23	4.23	.053
801104	55702		2CA3	40.35	88.23	8.63	.062
801104	55703		2CA3	40.35	88.23	0.05	.035
801105	55704		2CA2	41.36	89.16	8.14	.001
801105	55705		2CA2	41.36	89.16	6.13	.062

For the example we are using, the variables are:

Variables	Type	Length	Comments
DATE	I		Collection date
SNUM	I		Sample number
CRSID	A	5	Cruise identification
STATID	A	10	Station identifier
LAT	R		
LONG	R		
DRYWT	R		
WETWT	R		
PBCONC	R		
MGCONC	R		

Two sessions are displayed. In the first session the following actions are taken:

1) build data base ANALYT
2) build format file FMT1 for DATE, SNUM, STATID, LAT, LONG
3) build format file FMT2 for SNUM, PBCONC, MGCONC
4) add all rows to data base with variables DATE, SNUM, STATID, LAT, LONG
5) add (using INSERT) PBCONC and MGCONC to rows created in 4).

The second session is a sample analysis session. A new variable is created and a subset is taken. For this subset the following actions are taken:

1) display values on terminals
2) calculate descriptive statistics
3) save all variables on disk file
4) build SAS job for submission to remote site

SESSION 1

```
DATMAN
WELCOME TO DATMAN -- VERSION 3.0
SPECIAL COMMANDS ARE:
    HELP   LIST CURRENT OPTIONS AND SOME DESCRIPTIONS
    TERM   USE WHENEVER TERMINAL INPUT IS DESIRED
    CLEAR  USE WHENEVER DESIRE TO ERASE ALL INPUT TO THAT POINT

ENTER COMMAND
* BLDDB

INPUT NAME OF DATA BASE TO BE CREATED.
THE NAME SHOULD BE FULLY QUALIFIED
* ANALYT
IF THERE IS A FILE WITH MOST OF THE NAMES YOU WISH
TO ENTER, PLEASE GIVE ITS NAME NOW
OTHERWISE ENTER CARRIAGE RETURN
*
NOW INPUT THE VARIABLE NAMES.  AFTER EACH NAME HIT CARRIAGE RETURN.
DO NOT WORRY ABOUT ERRORS NOW YOU WILL BE GIVEN A CHANCE TO MAKE CORRECTION
INDICATE END OF LIST OF NAMES BY CR
* DATE
* SNUM
* CRSID
* STATID
* LAT
* LONG
* PBCONC
* MGCONC
*
EACH VARIABLE NAME WILL NOW BE DISPLAYED.
IF IT IS CORRECT THEN HIT CARRIAGE RETURN.
IF YOU WISH TO DELETE THE NAME THEN TYPE KILL.
IF YOU WISH TO CHANGE THE NAME THEN TYPE THE CORRECTED NAME
DATE *
SNUM *
CRSID *
STATID *
LAT *
LONG *
```

```
PBCONC *
MGCONC *
ENTER ANY ADDITIONAL NAMES NOW
CARRIAGE RETURN ENDS THE LIST
*
NOW FOR EACH VARIABLE, ENTER THE TYPE
THE VALID TYPES ARE --
A - FOR CHARACTER VARIABLES
I - FOR INTEGER VARIABLES
R - FOR REAL VARIABLES
DATE* I
SNUM* I
CRSID* A
STATID* A
LAT* R
LONG* R
PBCONC* R
MGCONC* R
FOR EACH ALPHABETIC VARIABLE INPUT THE NUMBER OF COLUMNS
THE DATA ITEMS WILL OCCUPY
CRSID * 5
STATID * 10
YOUR DATA BASE IS 105 COLUMNS LONG
HOW MANY SPACES DO YOU WISH TO SAVE FOR EXPANSION
* 50
ENTER NAMES OF THE VARIABLES WHICH WILL UNIQUELY IDENTIFY A CASE
INPUT CARRIAGE RETURN WHEN THE LIST IS DONE
* SNUM
*
INPUT NAME OF THE SECONDARY INDEX VARIABLES.
THE VARIABLES USED TO UNIQUELY IDENTIFY ARE ALREADY INDEX VARS
INPUT CR WHEN THIS LIST IS DONE.  YOU DO NOT NEED ANY OF THESE
THEY ARE USED FOR SPEEDING UP THE SEARCH PROCEDURE
* STATID
*
FOR EACH VARIABLE ENTER DESCRIPTIVE COMMENTS.
NO MORE THAN ONE LINE FOR EACH VARIABLE
DATE
* COLLECTION DATE (YYMMDD)
SNUM
* SAMPLE NUMBER
CRSID
* CRUISE IDENTIFIER
STATID
* STATION IDENTIFIER
LAT
* STATION LATITUDE
LONG
* STATION LONGITUDE
PBCONC
* CONCENTRATION OF LEAD
MGCONC
* CONCENTRATION OF MAGNESIUM
```

```
(re-enter DATMAN automatically)
ENTER COMMAND
* BLDFMT
INPUT THE NAME OF THE FORMAT FILE WHICH IS TO BE CREATED
* FMT1
IF THERE IS A FILE WITH MOST OF THE NAMES YOU WISH
TO ENTER, PLEASE GIVE ITS NAME NOW
OTHERWISE ENTER CARRIAGE RETURN
*
NOW INPUT THE VARIABLE NAMES.  AFTER EACH NAME HIT CARRIAGE RETURN.
DO NOT WORRY ABOUT ERRORS NOW YOU WILL BE GIVEN A CHANCE TO MAKE CORRECTIONS
INDICATE END OF LIST OF NAMES BY CR
* DATE
* SNUM
* LATE
* LONG
*
EACH VARIABLE NAME WILL NOW BE DISPLAYED.
IF IT IS CORRECT THEN HIT CARRIAGE RETURN.
IF YOU WISH TO DELETE THE NAME THEN TYPE KILL.
IF YOU WISH TO CHANGE THE NAME THEN TYPE THE CORRECTED NAME
DATE *
SNUM *
STATID *
LATE * LAT
LAT *
LONG *
ENTER ANY ADDITIONAL NAMES NOW
CARRIAGE RETURN ENDS THE LIST
*
NOW FOR EACH VARIABLE, ENTER THE TYPE
THE VALID TYPES ARE --
A - FOR CHARACTER VARIABLES
I - FOR INTEGER VARIABLES
R - FOR REAL VARIABLES
DATE* I
SNUM* I
STATID* A
LAT* R
LONG* R
IS THIS FORMAT ONLY FOR INTERACTIVE INPUT(Y/N)
* Y

(re-enter DATMAN automatically)
ENTER COMMAND
* BLDFMT
INPUT THE NAME OF THE FORMAT FILE WHICH IS TO BE CREATED
*FMT2
IF THERE IS A FILE WITH MOST OF THE NAMES YOU WISH
TO ENTER, PLEASE GIVE ITS NAME NOW
OTHERWISE ENTER CARRIAGE RETURN
*
NOW INPUT THE VARIABLE NAMES.  AFTER EACH NAME HIT CARRIAGE RETURN.
DO NOT WORRY ABOUT ERRORS NOW YOU WILL BE GIVEN A CHANCE TO MAKE CORRECTIONS
INDICATE END OF LIST OF NAMES BY CR
```

```
* SNUM
* PBCONC
* MGCONC
*
EACH VARIABLE NAME WILL NOW BE DISPLAYED
IF IT IS CORRECT THEN HIT CARRIAGE RETURN.
IF YOU WISH TO DELETE THE NAME THEN TYPE KILL.
IF YOU WISH TO CHANGE THE NAME THEN TYPE THE CORRECTED NAME
SNUM *
PBCONC *
MGCONC *
ENTER ANY ADDITIONAL NAMES NOW
CARRIAGE RETURN ENDS THE LIST
*
NOW FOR EACH VARIABLE, ENTER THE TYPE
THE VALID TYPES ARE --
A - FOR CHARACTER VARIABLES
I - FOR INTEGER VARIABLES
R - FOR REAL VARIABLES
SNUM* I
PBCONC* R
MGCONC* R
IS THIS FORMAT ONLY FOR INTERACTIVE INPUT(Y/N)
* Y

(re-enter DATMAN automatically)
ENTER COMMAND
* ADD

ENTER NAME OF DATA BASE
* ANALYT

ENTER NAME OF INPUT FILE
* TERM

PLEASE ENTER NAME OF THE FORMAT FILE
* FMT1

ENTER OUTPUT FILE NAME (CR GETS NO OUTPUT)
*
VERIFY COMMAND FILE IS OKAY
ADD          ANALYT
INPUT        TERM
FORMAT       FMT1
OUTPUT

ANY CORRECTIONS (YES OR NO)? * NO
ENTER NAME OF COMMAND SAVE FILE.  IF NONE, ENTER CR. *
THE INPUT WILL COME FROM THE TERMINAL
TYPE @@ WHEN YOU ARE DONE WITH THE INPUT
IF YOU WISH TO VERIFY THE INPUT AFTER EACH RECORD
TYPE YES NOW, OTHERWISE HIT CARRIAGE RETURN* YES
```

```
DATE
* 801020
SNUM
* 85602
STATID
* ICEA
LAT
* 42.365
LONG
* 85.62
FOR EACH VARIABLE, HIT CR IF CORRECT
ENTER NEW VALUE IF NOT CORRECT
DATE:          801020
*
SNUM:           85602
*
STATID:   ICEA
*
LAT:         42.3650000
*
LONG:        85.6200000
*
(input for additonal rows not displayed)
DATE
* 801105
SNUM
* 55705
STATID
* 2CA3
LAT
* 41.36
LONG
* 89.16
FOR EACH VARIABLE, HIT CR IF CORRECT
ENTER NEW VALUE IF NOT CORRECT
DATE:          801105
*
SNUM:           55705
*
STATID:         2CA2
*
LAT:       41.3600000
*
LONG:      89.1600000
*
DATE
*@@

(re-enter DATMAN)
ENTER COMMAND
* INSERT

ENTER NAME OF DATABASE
* ANALYT
```

```
ENTER NAME OF THE INPUT FILE
* TERM

PLEASE ENTER NAME OF THE FORMAT FILE
* FMT2

INPUT VERIFICATION VARIABLES
*
ENTER OUTPUT FILE NAME (CR GETS NO OUTPUT)
*
VERIFY COMMAND FILE IS OKAY
INSERT      ANALYT
INPUT       TERM
FORMAT      FMT2
VER
OUTPUT

ANY CORRECTIONS (YES OR NO)? * NO
ENTER NAME OF COMMAND SAVE FILE.  IF NONE, ENTER CR. *
THE INPUT WILL COME FROM THE TERMINAL
TYPE @@ WHEN YOU ARE DONE WITH THE INPUT
IF YOU WISH TO VERIFY THE INPUT AFTER EACH RECORD
TYPE YES NOW, OTHERWISE HIT CARRIAGE RETURN*

SNUM
* 85602
PBCONC
* 3.56
MGCONC
*
    1 RECORDS CHANGED

(input for additional rows not displayed)

SNUM
* 55705
PBCONC
* 6.13
MGCONC
* .062
    1 RECORDS CHANGED

SNUM
* @@

(re-enter DATMAN)
ENTER COMMAND
* STOP
```

SESSION 2

```
DATMAN
WELCOME TO DATMAN -- VERSION 3.0
SPECIAL COMMANDS ARE:
    HELP   LIST CURRENT OPTIONS AND SOME DESCRIPTIONS
    TERM   USE WHENEVER TERMINAL INPUT IS DESIRED
    CLEAR  USE WHENEVER DESIRE TO ERASE ALL INPUT TO THAT POINT

ENTER COMMAND
* SEARCH

ENTER NAME(S) OF DATA BASES TO BE SEARCHED
* ANALYT

ENTER NAME OF FILE WHERE SEARCH INPUT IS TO COME FROM (CR GETS TERMINAL)
*
VERIFY COMMAND FILE IS OKAY
SEARCH      ANALYT
INPUT

ANY CORRECTIONS (YES OR NO)? * NO
ENTER NAME OF COMMAND SAVE FILE.  IF NONE, ENTER CR. *
BEGIN INPUT TO SEARCH PROCEDURE
TYPE STOP IF YOU WISH TO HALT THE SEARCH
* A = PBCONC/MGCONC
* IF(STATID EQ 'ICEA')A = 1.5
* INCLUDE IF SNUM > 55000

      7 RECORDS WERE ACCEPTED

ENTER COMMAND
* DISPLAY SNUM,STATID,A,PBCONC,MGCONC

    SNUM     STATID                A          PBCONC          MGCONC
     85602   ICEA              1.5000000      3.5600000
     85603   ICEA              1.5000000      6.2800000
     55701   2CA3             79.8113208      4.2300000       0.0530000
     55702   2CA3            139.1935484      8.6300000       0.0620000
     55703   2CA3              1.4285714      0.0500000       0.0350000
     55704   2CA2           8140.0000000      8.1400000       0.0010000
     55705   2CA2             98.8709677      6.1300000       0.0620000
LAST RECORD, HIT CARRIAGE RETURN WHEN DONE*

ENTER COMMAND
* MEANS

ENTER VARIABLE NAMES FOR WHICH MEANS ARE TO BE TAKEN
* A,MGCONC,PBCONC

ENTER VARIABLE NAMES FOR CATEGORIZATION
CR IF ONLY WANT OVERALL MEANS
* STATID

ENTER OUTPUT FILE (CR IF NO OUTPUT)
* ON TERM
```

```
ENTER SAVE DATA BASE AND OPTIONS (CR IF NO SAVING TO BE DONE)
*
VERIFY COMMAND FILE IS OKAY
VAR          A,MGCONC,PBCONC
CATEG        STATID
OUTPUT       ON TERM
SAVE

ANY CORRECTIONS (YES OR NO)? * NO
ENTER NAME OF COMMAND SAVE FILE.  IF NONE, ENTER CR. *
  STATID      VARIABLE  NUM     MEANS        STD DEV        MAX        MIN
   2CA2         A         2   4119.43457      0.0000     8140.000    98.870
                MGCONC    2      0.03150      0.0000        0.062     0.001
                PBCONC    2      7.13499      0.0000        8.139     6.129

   2CA3         A         3     73.47780     69.1005      139.193     1.428
                MGCONC    3      0.05000      0.0137        0.062     0.035
                PBCONC    3      4.30333      4.2904        8.629     0.050

   ICEA         A         2      1.50000      0.0000        1.500     1.500
                MGCONC    0
                PBCONC    2      4.91999      0.0000        6.279     3.559

OVERALL MEANS
                A         7   1208.90011   3056.8238     8140.000     1.428
                MGCONC    5      0.04260      0.0257        0.062     0.001
                PBCONC    7      5.28856      2.9591        8.629     0.050

ENTER COMMAND
* SAVE ON TEMP *        (* is wild card for all variables)

ENTER COMMAND
* DATMAN

WELCOME TO DATMAN -- VERSION 3.0
SPECIAL COMMANDS ARE:
    HELP   LIST CURRENT OPTIONS AND SOME DESCRIPTIONS
    TERM   USE WHENEVER TERMINAL INPUT IS DESIRED
    CLEAR  USE WHENEVER DESIRE TO ERASE ALL INPUT TO THAT POINT

ENTER COMMAND
* SAS

ENTER YOUR NAME
* DATMAN

INPUT NAME OF SEARCHABLE FILE.  CR DEFAULTS TO DATKEEP
* TEMP

ENTER JOB OUTPUT FILE (OUTPUT OPTION)
*

INPUT NAME OF JCL FILE
* SASJCL
```

```
ENTER FILE NAME FOR SAS COMMANDS
* TERM
VERIFY COMMAND FILE IS OKAY
SAS          DATMAN
INFIL        TEMP
OUTPUT
JCL          SASJCL
SASPROG      TERM

ANY CORRECTIONS (YES OR NO)? * NO
ENTER NAME OF COMMAND SAVE FILE.  IF NONE, ENTER CR. *
ENTER SAS PROGRAM.  CR INDICATES END OF SAS INPUT
* THIS IS SAS PROGRAM CARD 1
* THIS IS SAS PROGRAM CARD 2
*

WELCOME TO DATMAN -- VERSION 3.0
SPECIAL COMMANDS ARE:
    HELP   LIST CURRENT OPTIONS AND SOME DESCRIPTIONS
    TERM   USE WHENEVER TERMINAL INPUT IS DESIRED
    CLEAR  USE WHENEVER DESIRE TO ERASE ALL INPUT TO THAT POINT

ENTER COMMAND
* STOP
```

APPENDIX B

DATA STRUCTURE OF DATA BASE AND NAMES DATA BASE

The structure of one logical record in a data base will be discussed. Since each logical record in a data base is independent of any other logical record, they are all of the same structure.

A logical record in a data base may consist of one or more physical records. The maximum physical record size used in DATMAN is 1000 words. The actual physical record size used is determined by the logical record size. If the logical record size is less than or equal to 1000 then this is the actual physical record size. If the logical record size is larger than 1000 then the number of physical records required per logical record is determined by dividing the logical record size by 1000 and rounding up. Once the number of physical records per logical record has been determined the physical record size is calculated by requiring all physical records to be the same size. Thus, the physical record size can be determined by dividing the logical record size by the number of physical records per logical record and rounding up.

A variable value within a logical record is determined by:

1) physical record number within logical record
2) low column of variable
3) high column of variable

DATMAN places and extracts values within column boundaries of the physical records. Integer values take up 4 columns, real values 8 bytes and alpha variables the length specified during creation of the data base.

The variables of a data base are controlled by a data base called nameDD where name is the name of the original data base. The variables of nameDD are:

1) NAMES - variable name - alpha length 10
2) COL1 - low column of variable - INTEGER
3) COL2 - high column of variable - INTEGER
4) INDCOD - index code - INTEGER
   - 0 - not index value
   - 1 - primary index
   - 2 - secondary index
5) TYPE - type of variable - INTEGER
   - 1 - integer (4 bytes)
   - 2 - alpha (COL2-COL1+1 bytes)
   - 3 - real (8 bytes)
6) COMMENT - description of variable - alpha length 80
7) LRECL - physical record number within one logical record - INTEGER

To add a new variable to data base name use ADD on data base nameDD and furnish values. To delete a variable use DELETE on nameDD with LOWVAL and HGHVAL specified in terms of NAMES. To change a characteristic of a variable use CHANGE with rows specified in terms of NAMES.

CAUTION

It is possible to destroy integrity of a data base by improper modification of the names data base. Some specific items of which to beware are:

1. Changing type of variable. Numeric values are stored in binary and hence any data in a numeric field would be invalid if interpreted as characters or vice versa. Numeric values are also interpreted as either integer or real depending upon their assigned type and any values would be incorrectly evaluated if the wrong type were chosen.
2. Changing column values of existing variables. This would result in incorrect values for those variables.
3. Changing any items of an index variable or making a new variable an index variable. If any changes are made to any index value (e.g., change of index code) then the terminal interface command INDEX must be run to reconstruct the index files.

APPENDIX C - VALID COMMANDS

The following commands are valid DATMAN commands:

ADD	add new records to data base
CHANGE	change existing values in data base
INSERT	add new data into existing rows in data base
SEARCH	retrieve data from data base
INDEX	create index files for data base
BMD	create a BMD job for submission to IBM site
SAS	create SAS job for submission to IBM site
NAMES	list valid names for data base on line printer
BLDDB	create a new data base
BLDFMT	create a format file for a fixed format, card image file
STOP	terminate execution
BLDSF	build a file with format file and data file for search and analysis
MEAN	calculate means and standard deviations (descriptive stat)
DELETE	delete records from a data base

The following commands are valid after SEARCH:

STOP	terminate this execution of DATMAN
DISPLAY	display values on terminal
LIST	place output on line printer
SAVE	save variables on disk file
SORT	sort data file by values of variables
DATMAN	reenter DATMAN for further processing
SEARCH	retrieve from either current subset or different data base
MEANS	take MEANS on current data base
MORE	make mode interactive for further processing
INPUT	give a file name which contains further commands.